# Real-Time Estimation of Multiple Potential Contact Locations and Forces

Dmitry Popov, Alexandr Klimchik, Anatol Pashkevich

# Real-Time Estimation of Multiple Potential Contact Locations and Forces

Dmitry Popov[1,2], Alexandr Klimchik[1], Anatol Pashkevich[2]

***Abstract*—In this work, we propose a contact point localization and external force estimation algorithm for collaborative robots. In comparison with existing approaches, the proposed algorithm can detect and evaluate multiple solutions and more than 3x faster without loss of accuracy. To achieve real-time performance, a hierarchical robot representation and surface mesh preprocessing was used, allowing us to achieve a 50x speedup in a run-time, compared to checking all contact points. Mesh preprocessing includes two-step clustering in the space of vertices normals vectors and vertices positions. The localization method was tested in a simulated Kinova Jaco 2 and real KUKA iiwa LBR 14 collaborative robots. Our solution allows estimating the contact point on the robot surface with 2.3 cm average accuracy in a more than 600 Hz loop.**

***Index Terms*—Physical Human-Robot Interaction, Force and Tactile Sensing, Human-Robot Collaboration, Contact Modeling**

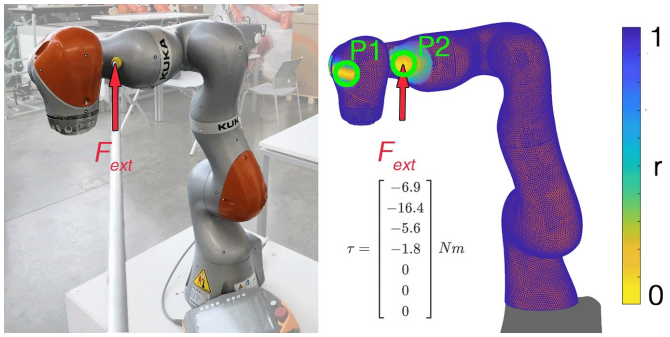



Fig. 1. Contact localization for the KUKA iiwa robot. Left: Robot pose and the actual point of contact. Right: estimated residual, collision in any of two potential contact points (P1 and P2) can produce the measured torque $\tau$.

## I. Introduction

COLLABORATIVE robots, especially manipulators are becoming more popular nowadays. One of their main advantages is the ability to work together with a human on the same task. Usually, this task is located in a shared environment between a robot and a human worker. As a result, a robot is forced to work in a complex environment with dynamic obstacles [1]. This will increase requirements for robot control to predict human intentions and actions. During this interaction, physical contact between a robot and a human could be unavoidable. However, the contact between a human body and a robot is not always a negative event. In some cases, physical interaction could be used to control the robot, program, or correct its path.

To ensure human safety in this human-robot collaboration, the interaction force should be estimated and controlled during the execution of the robot program. In addition, not only the magnitude of the force is important, but also its direction and position where it is applied. The time, required to estimate the contact point and the force is also critical. Since the robots can achieve high speeds, only a split second is enough to harm a human or the environment.

Our work is dedicated to creating a method for fast estimation of a contact point on the robot surface. We assume that the fact of the collision is already known by using any of the existing approaches [2], [3], [4] and the robot is equipped only with 1D joint torque sensors.

The problem of contact localization for collaborative robots was formulated in the past decade. The authors in [5] proposed some simplifications of the robot shape to formulate the problem by only two constrained parameters. Later this method was used in [6], [7] and compared with machine learning approaches. Manuelli and Tedrake [8] suggested using a particle filter for collision localization. In this approach, particles are projected on the robot surface, and for each particle probability of contact is estimated. The authors also introduced a friction cone constraint for the external force. In [9] the motion model of particles was modified with a combination of small self-movements of the robot to increase accuracy. Monte Carlo localization was used in [10] and compared to direct optimization and machine learning methods. In both works the authors estimated only the forces acting normal to the surface to decrease computational demand.

In one of our previous works [11] the contact isolation was done using two-step optimization on the mesh surface of the robot KUKA iiwa 14. In another work [12], we used a neural network trained to find a contact location in simulation and then transferred this network to the real robot.

Unfortunately, no of the existing approaches deals with the uncertainties of a localization algorithm, when only single-axis torque sensors are available. These methods are designed to output only one contact position. Depending on the robot pose and an external force direction, multiple solutions are possible as shown in Fig. 1. Torques, generated by one physical collision at P2, could be also described by another force at P1. So, in this particular configuration with estimated torques, the real collision could be in P1 or P2. In practice, the location

[1]The authors are with Center for Technologies in Robotics and Mechatronics Components and Institute of Robotics and Computer Vision, Innopolis University, Innopolis, Russia {d.popov, a.klimchik}@innopolis.ru

[2]The authors are with IMT-Atlantique Nantes, Nantes, France Laboratoire des Sciences du Numérique de Nantes (LS2N), Nantes, France anatol.pashkevich@imt-atlantique.fr

Digital Object Identifier (DOI): see top of this page.

of the collision will define the possible reaction of the robot. For example, we have a real contact with a human in the robot elbow which also could be represented by contact at the robot end-effector. The reaction, in the case of contact in the elbow, is to increase robot compliance and limit interaction force, but in the case of end-effector is to compensate external force, which are opposite reactions. Here, the choice of incorrect reaction could harm the human operator. Another example could be a collision-avoidance reaction which will be different for collision in elbow and end-effector.

In this work, we propose an algorithm that is capable of estimating all possible solutions, even if they are located on a different link. We present a robot mesh preprocessing procedure that allows to implement a real-time execution.

## II. Problem Statement

In the event of a collision between a robot arm and an external object, it is important to identify where exactly this collision happened. This information can be further used to chose a reaction action to ensure safety. In this work, we will address the solution of this problem for a serial manipulator, equipped with 1D torque sensors in the joints.

Lets us consider serial manipulator with a $n$ degrees of freedom. The robot state at rest can be described by the joint coordinate vector $\boldsymbol{q} \in \mathbb{R}^n$ and the joint torque vector $\boldsymbol{\tau} \in \mathbb{R}^n$. The robot has a collision in the point $\boldsymbol{p}_c \in \mathbb{R}^3$ with an external object and this point is located on the robot surface $\boldsymbol{p}_c \in \boldsymbol{C}$. This physical interaction leads to the external wrench $\boldsymbol{W}_{ext} \in \mathbb{R}^6$ applied in $\boldsymbol{p}_c$. Here $\boldsymbol{W}_{ext} = [\boldsymbol{F}_{ext}, \boldsymbol{M}_{ext}]^T$, where $\boldsymbol{F}_{ext} \in \mathbb{R}^3$ is an external force and $\boldsymbol{M}_{ext} \in \mathbb{R}^3$ is an external moment. For simplicity, the $\boldsymbol{\tau}$ values here caused only by the external wrench $\boldsymbol{W}_{ext}$.

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_c)^T \boldsymbol{W}_{ext} \tag{1}$$

In general, mapping between forces and torques in Cartesian and joint coordinate systems can be expressed with a robot kinematic Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}, \boldsymbol{p}_c) \in \mathbb{R}^{6\times n}$ where $\boldsymbol{p}$ is a point in the local coordinate frame of the link with $\boldsymbol{W}_{ext}$ (1). One column of a Jacobian matrix, can be found by:

$$\boldsymbol{J}_j = \begin{bmatrix} \boldsymbol{J}\boldsymbol{v}_j \\ \boldsymbol{J}\boldsymbol{w}_j \end{bmatrix} = \begin{bmatrix} \boldsymbol{Z}_{j-1} \times (\boldsymbol{P} - \boldsymbol{O}_{j-1}) \\ \boldsymbol{Z}_{j-1} \end{bmatrix} \tag{2}$$

where $\boldsymbol{J}_j$ is a $j$th column of a Jacobian matrix, $\boldsymbol{J}\boldsymbol{v}_j$ corresponds to linear, $\boldsymbol{J}\boldsymbol{w}_j$ to angular velocity Jacobian. $\boldsymbol{P}$ is a position of an arbitrary point $\boldsymbol{p} \in \boldsymbol{C}$ in the base coordinate frame, $\boldsymbol{O}_{j-1}$ is an origin of a previous joint and $\boldsymbol{Z}_{j-1}$ its axis of rotation. All of these values depend on $\boldsymbol{q}$.

In Eq. (1) $\boldsymbol{q}$, $\boldsymbol{\tau}$ are known from the robot state, and to solve a localization task we need to find $\boldsymbol{W}_{ext}$ and $\boldsymbol{p}_c$. The straightforward solution is to use a pseudo-inverse of the Jacobian $\boldsymbol{J}^{(i)}(\boldsymbol{q}, \boldsymbol{p})$ for all points on the robot surface $\boldsymbol{C}$. To do so, we fix the potential collision point $\boldsymbol{p}$ and a link $i$ where this point lies to compute Jacobian.

The problem can be simplified by the assumption of a point contact collision. This leads to $\boldsymbol{M}_{ext} = 0$ and it is sufficient to consider only $\boldsymbol{F}_{ext}$ component of the $\boldsymbol{W}_{ext}$. In this case Jacobian becomes $\boldsymbol{J}\boldsymbol{v}^{(i)}(\boldsymbol{q}, \boldsymbol{p_c}) \in \mathbb{R}^{3\times n}$. When $i = 3$ there will be exactly one solution for all point on the surface of $i$th link, but for the $i > 3$ we can find an approximate solution for an over-determined system by:

$$\min_{\boldsymbol{F},\ \boldsymbol{p}\in\boldsymbol{C}} \left\| \boldsymbol{\tau} - \boldsymbol{J}\boldsymbol{v}^{(\boldsymbol{i})}(\boldsymbol{q}, \boldsymbol{p})^T \boldsymbol{F} \right\| \tag{3}$$

Assuming that we also need to iterate through points on the curved robot surface $\boldsymbol{C}$, multiple solutions are possible. The example of these multiple minimum cases is shown in Fig. 1 where torque in joints can be described by the contact in point $P1$ or $P2$. This multiple minima behavior could be explained by the existence of point-force pairs that could generate torque equal to the observed from the robot sensors. In practice, it could lead to wrong external force and collision point estimation and as result wrong robot reaction to the collision. Thus, this behavior should be studied.

## III. Presence of Multiple Solutions

### A. Planar Robot Examples

To illustrate multiple solution cases, let us consider a planar 4 degree of freedom robot with link length $\boldsymbol{L} = [1, 1, 1, 1]$. The external force $\boldsymbol{F} \in \mathbb{R}^{2\times 1}$ in the contact point $\boldsymbol{p}$ and vector of external torque $\boldsymbol{\tau} \in \mathbb{R}^{4\times 1}$ caused by the original force $\boldsymbol{F}_{ext}$.

The solutions of (3) can be found by numerically estimating the residual $r(i, \boldsymbol{p})$ for points on the link surface $\boldsymbol{p}$:

$$r(i, \boldsymbol{p}) = \|\boldsymbol{\tau} - \boldsymbol{\tau_p}\| \tag{4}$$

where $\boldsymbol{\tau}_p$ found from (1) for a contact point $\boldsymbol{p}$ on the $i$-th link using pseudo-inverse. The residual value of 0 means exact recreation of input torque $\boldsymbol{\tau}$, while other values mean inability to find a force vector that will result in $\boldsymbol{\tau}$ torque. The presence of multiple solutions will result in multiple minima in $r(i, \boldsymbol{p})$.

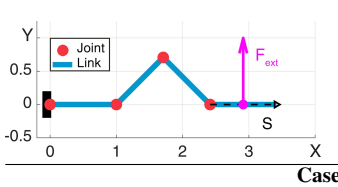
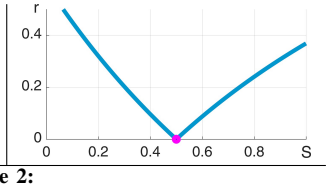
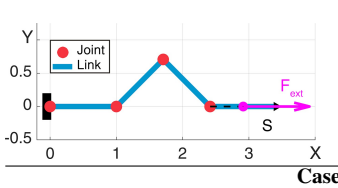
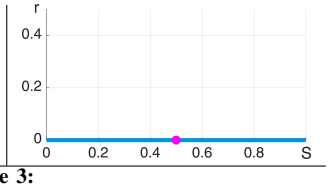
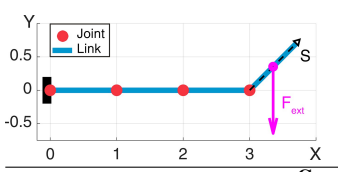
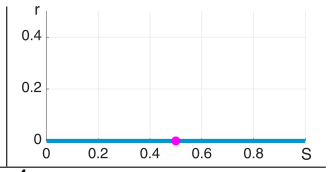
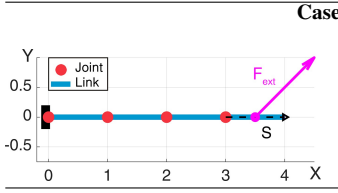
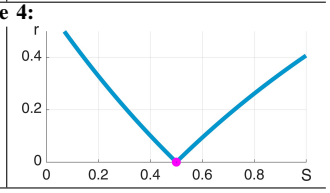
Let us examine few cases where multiple minima can be detected. At the first stage, consider a planar robot with rod links, where the contact point can be placed on the axis of the link. The summary of test cases is presented in Table I.

*1) Case 1:* The robot is in a configuration $\boldsymbol{q} = [0^\circ\ 45^\circ\ -90^\circ\ 45^\circ]^T$, has a collision in point $i = 4$ $\boldsymbol{p}_c = [0.5, 0]^T$, interaction force is $\boldsymbol{F}_{ext} = [0, 1]^T$. This robot configuration is not singular kinematically and from the external force point of view. As result, we have only one point with zero residual and one corresponding external force.

*2) Case 2:* $\boldsymbol{q} = [0^\circ\ 45^\circ\ -90^\circ\ 45^\circ]^T$, collision in $i = 4$ $\boldsymbol{p}_c = [0.5, 0]^T$, interaction force is $\boldsymbol{F}_{ext} = [1, 0]^T$. From the kinematic point of view, this configuration is not singular, but the direction of $\boldsymbol{F}_{ext}$ intersects the 4th joint axis and produces a zero torque $\tau_4$ making it singular to the given external force. As a result, any point on the 4th link can be used to describe an interaction. The estimated external force vector will be equal to the original contact force at any point of the 4th link.

*3) Case 3:* $\boldsymbol{q} = [0^\circ\ 0^\circ\ 0^\circ\ 45^\circ]^T$, collision in $i = 4$ $\boldsymbol{p}_c = [0.5, 0]^T$, interaction force is $\boldsymbol{F}_{ext} = [0, -1]^T$. The Jacobian rank $rank(\boldsymbol{J}_p) = 2$, this configuration is not kinematically singular, external force does not intersect joints axis, but has multiple minima. The main reason for singularity here is the position of all joint axis, where all of them lies on the line. In this situation, at any point of the 4th link, we can have an infinite number of forces that will result in a zero residual.

TABLE I
SINGULAR AND NON-SINGULAR CASES FOR ROD ROBOT MODEL.

| Robot configuration | Cost function $r$ eq. (4) |
|---|---|
| **Case 1:** | |
|  |  |
| **Case 2:** | |
|  |  |
| **Case 3:** | |
|  |  |
| **Case 4:** | |
|  |  |

*4) Case 4:* $\boldsymbol{q} = [0° \ 45° \ -90° \ 45°]^T$, collision in $i = 4$ $\boldsymbol{p}_c = [0.5, 0]^T$, interaction force is $\boldsymbol{F}_{ext} = [1, 1]^T$. Here the Jacobian rank $rank(\boldsymbol{J}_p) = 1$ and robot pose is kinematically singular, all joint centers lie on the line. This case has one point of collision and an infinite number of force vectors which can be applied to this point to achieve a zero residual.

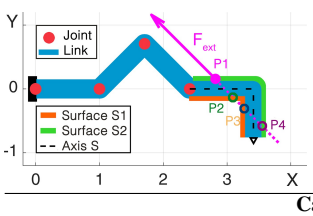
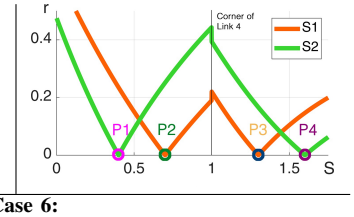
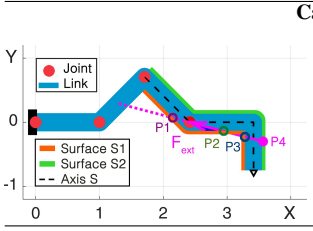
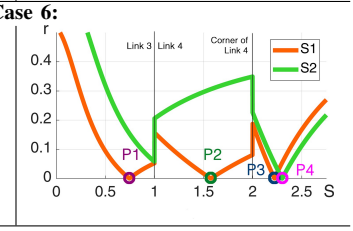
For before mentioned cases, the multiple minima behavior mostly observable in a singular configuration. Here the term singular corresponds to not only kinematic singularity but also for the instances with aligned joint centers and when the force intersects the last joint.

So far we observed multiple solutions in a rod link robot, but in the real world, the robot's link could have a complex shape. In addition, the external force could be applied only to the surface. The presence of a surface could increase the number of possible solutions even in non-singular cases.

Let us now consider the same 4 DoF planar robot with a thicker link, where the contact point can lie only on the right or left surface of the the $\Gamma$ shaped last link. The summary of test cases is presented in Table II.

*5) Case 5:* The robot with $\boldsymbol{q} = [0° \ 45° \ -90° \ 45°]^T$ has a collision in $i = 4$ $\boldsymbol{p}_c = [0.4, 0.15]^T$, the interaction force is $\boldsymbol{F}_{ext} = [-1, 1]^T$. The robot pose is not singular kinematically and from the external force point of view, since the force vector does not intersect the last joint axis. The line of force action intersects the 4th link surfaces in 4 points $P1, P2, P3, P4$, where $P1$ is an original point of contact. This scenario is similar to $Case$ 1, but the existence of a surface increased the number of potential solutions to 4. All potential

TABLE II
NON-SINGULAR CASES FOR ROBOT MODEL WITH SURFACE.

| Robot configuration | Cost function $r$ eq. (4) |
|---|---|
| **Case 5:** | |
|  |  |
| **Case 6:** | |
|  |  |

intersection points have a zero residual value. Therefore the same force could be applied in any of these points without any possibility to distinguish between them in static.

*6) Case 6:* $\boldsymbol{q} = [0° \ 45° \ -90° \ 45°]^T$, collision in $i = 4$ $\boldsymbol{p}_c = [1.15, -0.3]^T$, the force is $\boldsymbol{F}_{ext} = [1.15, 0.3]^T$. The robot is in the same configuration as in $Case$ 5, but the force vector now intersects the 4th joint axis and produces a zero torque in $\tau_4 = 0$. Points $P2, P3, P4$ lie on the 4th link and $P1$ on the 3rd. This case is similar to the $Case$ 2, but because of the link surface, we no longer have an infinite number of possible contact points. Now, it is impossible to distinguish even the link of the robot with a collision.

### B. Existing Approaches for Multiple Minima Resolution

Some of the current approaches in contact localization reduce the number of possible solutions by putting constraints on the direction of the external force. Assuming that there is physical contact between the robot and an object, there should be some friction forces defined by the friction coefficient. Thus, the external force will lie inside of a friction cone with a friction coefficient $\mu > 0$, directed by the normal of a robot surface. The eq. (3) can be supplemented with an additional condition $\boldsymbol{F} \in \boldsymbol{FC_p}$ where $\boldsymbol{FC_p}$ is a friction cone in point $\boldsymbol{p}$ defined by its normal vector $\boldsymbol{p}_n$ and a $\mu$. We can simplify this problem by approximation of friction cone with the tetrahedral pyramid $\boldsymbol{F} = \sum_{i=1}^{4} \alpha_i \boldsymbol{F}_{cone,i}$.

Here, $\alpha_i$ is the weight of each support vector $\boldsymbol{F}_{cone,i}$. It allows to rewrite the optimization problem (3) in terms of quadratic programming for a fixed contact point $\boldsymbol{p}$:

$$\min_{\boldsymbol{a} \geq 0} \left( \boldsymbol{a}^T \boldsymbol{H} \boldsymbol{a} + \boldsymbol{g}^T \right) \tag{5}$$

where $\boldsymbol{H} = \boldsymbol{J}_a \boldsymbol{J}_a^T$ is a square of Jacobians, variable $\boldsymbol{a}$ is weight for a support vectors $4 \times 1$, $\boldsymbol{g} = -\boldsymbol{J}_a \boldsymbol{\tau}_{ext}$. More information about this approach could be found in [8].

This allows us to reduce the number of solutions by eliminating points where the force is directed opposite to the normal vector or vice versa. For example, in $Case$ 5 it will drop out points $P2$ and $P4$. It should be noted that this constraint will

**Algorithm 1:** Link mesh clustering

**Function** `geoCluster`($\boldsymbol{v}$, $\boldsymbol{f}$, $num\_cl$)**:**
  **Input :** Mesh geometry ($\boldsymbol{v}$, $\boldsymbol{f}$), number of clusters $num\_cl$
  **Output:** List of clusters $\boldsymbol{c}$
  1st step: Farthest Point Sampling;
  $\boldsymbol{g} = random(\boldsymbol{v})$;
  **for** $1 : num\_cl$ **do**
    $\boldsymbol{D} = geoDist(\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{g})$;
    $n\_ind = find(max(\boldsymbol{D}))$;
    $\boldsymbol{g}.add(n\_ind)$;
  $\boldsymbol{c} = voronoi(\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{g})$;
  2nd step: Lloyd's Relaxation;
  **while** $\boldsymbol{c} \neq \boldsymbol{c\_prev}$ **do**
    **for** *each* $\boldsymbol{c}$ **do**
      $\boldsymbol{vi}, \boldsymbol{fi} = \boldsymbol{c}(\boldsymbol{v}, \boldsymbol{f})$;
      $\boldsymbol{g} = findGeoCenter(\boldsymbol{vi}, \boldsymbol{fi})$;
    $\boldsymbol{c} = \boldsymbol{c\_prev}$;
    $\boldsymbol{c} = voronoi(\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{g})$;

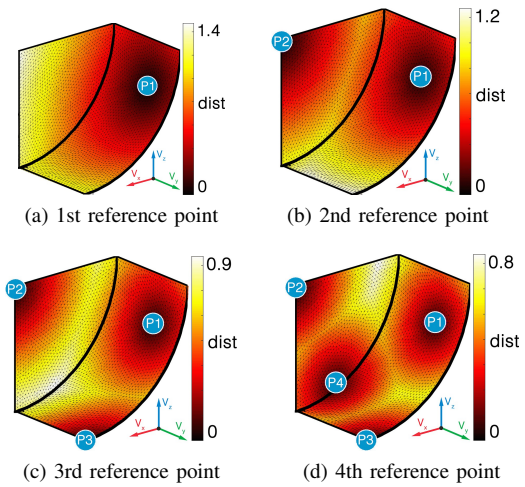
not help in the scenario where external force intersects the joint axis and a potential collision point could be on a different link. The approximation of the friction cone could also be a source of errors for the force vectors that lie inside of the friction cone, and outside of the pyramid.

## IV. Proposed Algorithms

In this section, we propose our way of finding a solution for the system (3). The system of the equation has two unknowns, $\boldsymbol{p}$ as a contact point and $\boldsymbol{F}$ as an external force. In particular, the point of contact $\boldsymbol{p}$ is constrained by the known surface $\boldsymbol{C}$ and an external force $\boldsymbol{F}$ by the normal vector in the contact point $\boldsymbol{p_n}$ and a friction coefficient $\mu$. Since the robot surface is usually formulated as a triangular mesh, it already includes information about link surface and link normal vectors. Thus, by analyzing the robot mesh, one can reduce the search space of the problem and as result reduce computational complexity, enabling real-time execution. However, robot mesh is mostly used for visualization purposes, therefore should be modified.

One way to avoid iteration through all points on the link surface is to introduce a tree-like structure. The idea behind that is to divide the surface into segments, each segment will be represented by the reference point. One significant requirement for the reference point and all segment points is that they should have similar properties in a sense of position and orientation. According to that, points in one segment should be clustered by their position and their normals orientation, so classical 3D clustering algorithms like k-means are not applicable. In this way, the robot will be the root of a tree, the first layer of leaves are links, each link has leaves as segments and each segment has bottom leaves as points on the surface. By isolating the link with a potential collision we evaluate the residual (4) in the reference points of this link a choice segments with the lowest residual value and finally search contact point only in these segments. As result, we greatly decrease the number of evaluations, but only if we chose the right segment.

The problem of multiple solutions is solved by estimating interaction force in the last link with non-zero torque in its joint and then recursively checking for an intersection between this force line of action and the next joint axis. If the line of force action intersects the axis, then this force could also appear on the next link as was shown in Case 6.




(a) 1st reference point (b) 2nd reference point

(c) 3rd reference point (d) 4th reference point

Fig. 2. The evolution of the Farthest Sampling Point algorithm. The color shows the geodesic distance (the distance on the surface of the mesh) between reference points and all mesh vertices.

### A. Surface Clustering

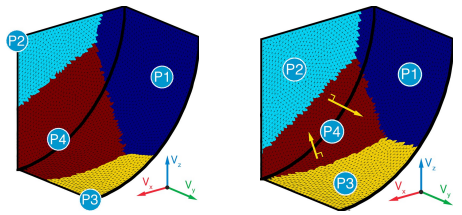
Let us assume that surface is described by a discrete triangular mesh which consists of a finite number of vertices $\boldsymbol{v} \in \mathbb{R}^{3\times m}$ and a list of facets $\boldsymbol{f} \in \mathbb{R}^{3\times n}$. Each facet consists of three edges between three points from $\boldsymbol{v}$. In addition, each vertex has a normal vector $\boldsymbol{n} \in \mathbb{R}^{3\times m}$, that locally follows the surface normal.

The segmentation or clustering allows to group a large number of vertices and represents them by one reference point. The first approach for making $k$ clusters is to sample $k$ uniformly distributed points on the robot surface and then use a Voronoi segmentation for all vertices. This can be achieved by the Farthest Point Sampling algorithm and following Lloyd's relaxation. The algorithm for this is presented in Algorithm 1. Now, let us describe this method in more detail.

Suppose there is a triangular mesh that we need to divide into four segments. The first step is to sample the first segment reference point by randomly choosing it from the vertices list and then find a distance to all other vertices $\boldsymbol{v}$ (Fig. 2a). The distance function here is not a Euclidean distance, but a geodesic distance that shows the distance on the mesh surface. In our algorithm, it implemented using the heat method [13] as a part of *gptoolbox* [14].

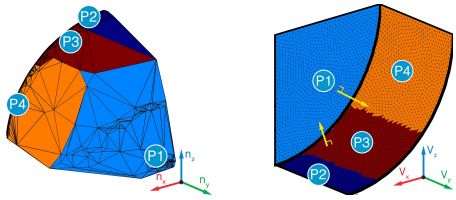
In the second step after measuring a distance to all points, select the farthest one as a second cluster reference point (Fig. 2b). Now the distance to some arbitrary point $\boldsymbol{p}$ on the surface can be estimated as minimum distance from existing reference points $dist = min(geoDist(P1, p), geoDist(P2, p))$. After that we repeat until the number of segments reference points is equal to the desired number of clusters (Fig. 2c,d).

The next step is to apply the Voronoi partition and assign a cluster to all points on the surface. It can be done to any surface point by finding the closest reference point and assigning the same class. The clustering result is shown in Fig. 3a. Since the reference points were fixed during the previous step, obtained




(a) Initial partition (b) After relaxation

Fig. 3. Clusters normalization. (a) shows segments for the reference points located as in Fig. 2d, clusters size is unbalanced, reference points are static. (b) shows segments after Lloyd's relaxation, where reference points iteratively moved to the center of each cluster.




(a) Clustering in $\boldsymbol{n}$ (b) Mapping to $\boldsymbol{v}$

Fig. 4. Clusters in a normal vector space. Applying farthest Point Sampling and Lloyd's relaxation for $\boldsymbol{n} = (n_x,\ n_y,\ n_z)$ (a) and projecting assigned clusters into Cartesian space for $\boldsymbol{v} = (v_x,\ v_y,\ v_z)$ (b).

clusters are unbalanced, reference points sometimes located in the corners of its segments ($P2$ and $P3$).

To balance the clusters and move the reference points we apply Lloyd's relaxation. By iteratively moving the reference points to the center of its segments, we can acquire balanced segments with almost the same size and reference points in their centers (Fig. 3b). It should be noted that the choice of initial point does not show any significant difference in final cluster size or quality.

If we look at the first and the fourth cluster with reference points $P1,\ P4$, they include two kinds of points: one of them lies on the vertical wall of a mesh and the second on the curved surface. Vertices normals in these two groups will be orthogonal, thus the reference point is capable of describing a position of neighborhood vertices, but not able to describe their orientation or normals. Although it is possible to increase the number of clusters, it will not guarantee that two points in the same cluster will have orthogonal normal vectors.

To overcome this problem, we can repeat the same steps of Algorithm 1, but for the normal vectors $\boldsymbol{n}$. The result for cluster segmentation in a normal vectors space is shown in Fig. 4a. By mapping clusters back to the vertices positions $\boldsymbol{v}$, the clusters are now grouped according to their normals, all vertices from the vertical wall are in the same cluster and a curved surface is divided by other three clusters (Fig. 4b).

### B. Robot Mesh Preprocessing

The robot mesh preprocessing procedure is executed offline and allows to obtain a tree-like structure for each link. In this procedure, all robot link meshes are clusterized in a similar way that was described in a previous section.

Usually, link mesh files are derived in a *.stl* format with a list of vertices coordinates $\boldsymbol{v}$, a list of facets $\boldsymbol{f}$, and a list of normal vectors for each facet. The image of such mesh for the second link of the KUKA iiwa robot is shown in Fig. 5a.

**Algorithm 2:** Mesh preprocessing

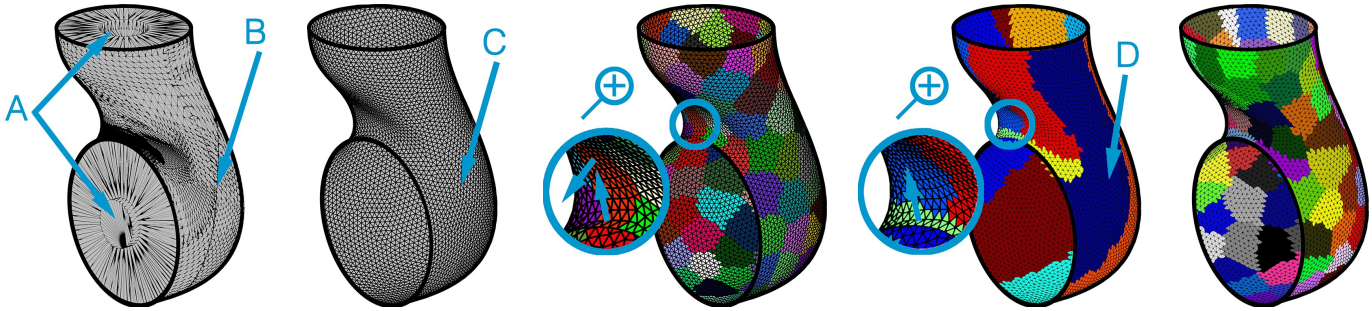
**Function** `clusterMesh` $(file)$**:**
  **Input :** Mesh file in *.stl* format
  **Output:** Data structure $\boldsymbol{link}$
  $d\_size$ desired size of a cluster;
  $\boldsymbol{v\_stl}, \boldsymbol{f\_stl},\ \boldsymbol{fn\_stl} = importSTL(file)$;
  $\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{fn} = isoRemesh(\boldsymbol{v\_stl}, \boldsymbol{f\_stl}, \boldsymbol{fn\_stl})$;
  $\boldsymbol{vn} = calcNormals(\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{fn})$;
  $\boldsymbol{cn} = geoCluster(\boldsymbol{vn}, \boldsymbol{f}, n\_cl)$;
  **for** *each* $\boldsymbol{cn}$ **do**
    $\boldsymbol{c\_i} = connectedComponents(\boldsymbol{cn})$;
    **for** *each* $\boldsymbol{c\_i}$ **do**
      **if** $size(\boldsymbol{c\_i} \geq d\_size)$ **then**
        $n\_cl = \lfloor size(\boldsymbol{v}(\boldsymbol{c\_i}))/d\_size \rfloor$;
        $\boldsymbol{c} = geoCluster(\boldsymbol{v}(\boldsymbol{c\_i}), \boldsymbol{f}(\boldsymbol{c\_i}), n\_cl)$;
      **else**
        $\boldsymbol{c} = \boldsymbol{c\_i}$;
      store $\boldsymbol{c}$ data in a $\boldsymbol{link}$;

The standard optimization routine for localization assumes a continuous function for the potential contact point $\boldsymbol{p}$, so in the case of a triangular mesh, it means that the contact point could be in a vertex or a facet of a mesh. The searching process implies a gradient descent-like method to identify the exact location of the local minimum on the surface $\boldsymbol{C}$ and a lot of computational power. At the same time, the noise in external torque measurements will limit the accuracy of localization. Thus, in our approach, we scatter a set of equally distributed points on the robot surface and estimate contacts only in these points. The average distance between scattered points is set to a value, which is several times lower than the desired localization accuracy. To obtain a new mesh with equally scattered vertices, we used an isotropic remesh algorithm from [15]. In addition to remeshing, the original link mesh file (Fig. 5a) was manually edited to remove internal areas of the link (A). The resulting mesh is presented in Fig. 5b. The area marked with (B) had very inconsistent triangles sizes compared to the same region of remeshed link (C).

Mesh clustering could be done using Algorithm 1 for a vertices coordinates $\boldsymbol{v}$. The output of this algorithm is shown in Fig. 5c. Again, as was described previously, two points of the same segment (shown in the zoomed image of a Fig. 5c) have almost orthogonal normals directions. This will result in a bad representation of a cluster with a reference point. To overcome this, Algorithm 1 was used in a vertices normals space $\boldsymbol{n}$ (Fig. 5d). The same zoomed region now shows that there are two separate clusters. Despite that, obtained clusters are unbalanced in size (D) and can have unconnected parts. Therefore we propose to use two-step clustering.

The stages of a two-step clustering, used for the link mesh preprocessing are demonstrated in Algorithm 2. The first step is the clustering of a link in a vertices normals space, and after that, clustering of all obtained clusters on subclusters. In this way, we achieve clusters of the desired size and approximately the same orientation. The result could be evaluated in Fig. 5e.

Compared to traditional clustering with a weighted objective function that considers vertices position and normals, our approach better describes small details. In traditional clustering




(a) Initial triangular link mesh imported from $.stl$.

(b) Isotropic remesh of the (a)

(c) Clustering result for vertices coordinates $(v_x,\ v_y,\ v_z)$

(d) Clustering result for vertices normals $(n_x,\ n_y,\ n_z)$

(e) Two step clustering

Fig. 5. Link mesh clustering steps. Imported mesh (a) from $.stl$ model have unreachable areas (A) and inconsistent vertex distribution (D). This mesh is preprocessed (b) by removing unnecessary areas and remeshed by uniformly placed vertex points (C). In (c) we applied Algorithm 1 for vertices coordinates, zoom-up image shows that two points in the same segment have almost perpendicular normals. We can avoid this by applying the same algorithm to vertices normals (d), but one can expect clusters with inconsistent size (D) and unconnected areas. As an alternative, a two-step clustering algorithm (e) is proposed (Algorithm 2) which eliminates the disadvantages of (c, d).

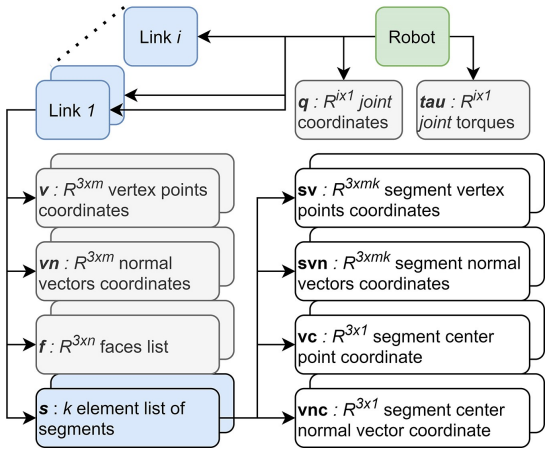



Fig. 6. Data structure for the robot state and internal surface representation.

with a weighted coefficient between vertices positions and normals (separated optimization is required to find optimal weights) will not give clusters with a fixed maximum size and most likely miss-classify minor parts of the surface. The two-stage algorithm generates clusters with limited maximum size to ensure algorithm run-time and is not limited with minimum size to preserve smaller surface elements and features.

After the segmentation of mesh vertices, data about each segment is stored in a ***Robot*** structure. Information about stored values is presented in Fig. 6.

### C. Localization Algorithm

In this section, we present an algorithm for collision point localization. In contrast with other available approaches, the algorithm is capable of estimating multiple solutions for the cases where the force vector intersects the last joint axis. The Algorithm 3 consists of two parts, the first part ($localization\_link$) deals with localization on the robot link, and the second part ($localization\_robot$) with the whole robot. Let us consider the first part in more detail.

As was mentioned previously, we try to exploit the proposed tree-like structure of the robot by estimating residual $\boldsymbol{r}$ in cluster reference points. For the reference points, their location in the global and local frame is known, the normal vector is given, joints position and torque are known from the robot state. Consequently, the external force vector $\boldsymbol{F}$ could be estimated using pseudo inversion. Typically there will be at least two points where the line of external force action intersects the link surface, but only one of them is possible due to friction cone constraint. This is why we manually set the residual as a large value for the points where an external force is out of the friction cone. It should be noted that instead of pseudo inversion we can solve a quadratic program (5) with a friction cone approximation, but it could decrease the performance of the algorithm.

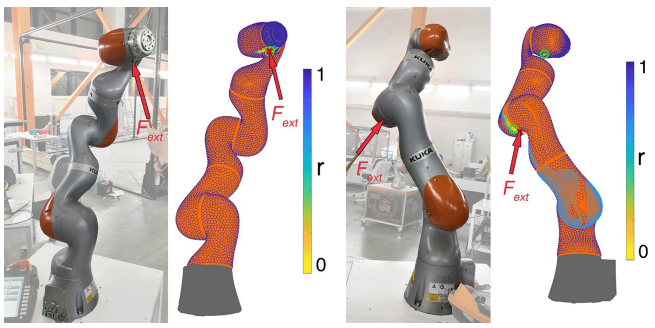
Next, we pick only $\boldsymbol{n}$ segments with the lowest $\boldsymbol{r}$ in their reference points and find the residual for the vertices of the segments. In this way, we avoid iterating through all link vertices and reduce computation requirements.

As the last stage, we need to check the possibility of multiple solutions. Multiple solutions on the same link are basically multiple minima on the already estimated residual $\boldsymbol{rp}$, so it is sufficient to output the local minima. However, the external force, estimated in the current link could intersect following after the link joint axis and generate zero torque in it. To address this issue we introduce a simple metric of a distance $\boldsymbol{d}$ between the estimated external force and the following joint axis. If there is an intersection, then we need to check the next link for a collision too.

The localization for the whole robot starts with link isolation. Firstly, the link following the last non-zero torque is isolated. In the practice, of course, there are some threshold values that depend on the joint torque noise. The isolated link is also could be extracted from the estimated collision location on the previous timestamp ($\boldsymbol{p\_prev}$) if there is one. This isolated link is checked for a collision in a $localization\_link$ routine and the following multiple solution evaluation. In this evaluation, we compare the distance between the estimated line of force action and the joint axis. In other words, if there is an intersection, we start to iteratively check the adjacent links until there will be no intersection, or we reach the last link. In this way, we can acquire multiple solutions on one

**Algorithm 3:** Localization on the robot surface

**Function** `estimate_F` ($ind$, $\boldsymbol{Robot}$):
  **Input :** Surface point index $ind$, structure $\boldsymbol{Robot}$
  **Output:** External force $\boldsymbol{F}$, residual $rp$
  $\boldsymbol{J} = Jac(Robot.Links(link).v(ind))$;
  $\boldsymbol{F} = (\boldsymbol{J}\ \boldsymbol{J}^T)^{-1}\ \boldsymbol{J}\ \boldsymbol{\tau_{ext}}$;
  **if** $\measuredangle(\boldsymbol{F}, Robot.Links(link).vn(ind)) < atan(\mu)$ **then**
    $\boldsymbol{\tau_t} = \boldsymbol{J}^T\boldsymbol{F}$;
    $\boldsymbol{rp} = \|\boldsymbol{\tau_{ext}} - \boldsymbol{\tau_t}\|$;
  **else**
    $\boldsymbol{rp} = Inf$;
**Function** `localization_link` ($link$, $\boldsymbol{Robot}$):
  **Input :** Link index $link$, Structure $\boldsymbol{Robot}$
  **Output:** Local collision point $\boldsymbol{p\_l}$, external force $\boldsymbol{F\_l}$, residual $\boldsymbol{r}$, dist. between $\boldsymbol{F\_l}$ and joint axis $\boldsymbol{d}$
  **for** *each* $Robot.Links(link).vc$ **do**
    $\boldsymbol{F},\ \boldsymbol{rp} = estimate\_F(ind, \boldsymbol{Robot})$;
  pick top $n$ clusters $\boldsymbol{c\_top}$ with min $\boldsymbol{rp}$;
  **for** *each* $\boldsymbol{c\_top}$ **do**
    **for** *each* $Robot.Links(link).c(c\_top).v$ **do**
      $\boldsymbol{F},\ \boldsymbol{rp} = estimate\_F(ind, \boldsymbol{Robot})$;
  $\boldsymbol{p\_l} = find(min(\boldsymbol{rp}))$;
  $\boldsymbol{F\_l} = \boldsymbol{F}(\boldsymbol{p\_l})$;
  check for joint sensor axis intersection;
  $\boldsymbol{O} = link + 1$-th joint axis;
  $\boldsymbol{d} = dist(\boldsymbol{F}, \boldsymbol{O})$;
**Function** `localization_robot` ($Robot$):
  **Input :** Structure $\boldsymbol{Robot}$, collision point on the previous timestamp $\boldsymbol{p\_prev}$
  **Output:** Collision point $\boldsymbol{p\_c}$, external force $\boldsymbol{F\_ext}$
  **if** $isEmpty(\boldsymbol{p\_prev})$ **then**
    isolate last link with nonzero torque;
    $link = lastNonZero(abs(\boldsymbol{\tau_{ext}}) > \boldsymbol{\epsilon})$;
  **else**
    $link =$ link with $\boldsymbol{p\_prev}$;
    $link.c =$ cluster with $\boldsymbol{p\_prev}$;
  $\boldsymbol{F},\ \boldsymbol{p},\ \boldsymbol{r},\ \boldsymbol{d} = localization\_link(link, \boldsymbol{Robot})$;
  **while** $abs(\boldsymbol{d}) < \boldsymbol{\epsilon_d}$ & $link < link\_max$ **do**
    $link++$;
    $\boldsymbol{F},\ \boldsymbol{p},\ \boldsymbol{r},\ \boldsymbol{d} = localization\_link(link, \boldsymbol{Robot})$;
  $\boldsymbol{p\_c} = find(min(\boldsymbol{r}))$;
  $\boldsymbol{F\_ext} = \boldsymbol{F}(\boldsymbol{p\_c})$;
  $\boldsymbol{p\_prev} = \boldsymbol{p\_c}$;




(a) Contact at the 6th link (b) Contact at the 4th link

Fig. 7. Localization algorithm for KUKA iiwa. Probability at all points is estimated for visualization. The green circle is a possible collision location, the red cross is a true contact location.

link and an additional solution from the following links.

Consider the example on the Fig. 7a. The KUKA iiwa robot has a collision on the 6th link. The last joint with non-zero torque is a joint 6, so we isolate the 6th link and try to estimate a contact point on it. There are two possible solutions, one of them is where the actual external force is applied and the second one is on the other side of the link which represents the same force but in the opposite direction. The second contact point is ignored due to friction cone constraint. The estimated force for the first potential contact point does not intersect the axis of a 7th joint. Thus, the robot in a given pose and interaction has only one solution.

Another example from the Fig. 7b shows multiple minima behavior. The real contact is on the 4th link and a 4th joint is the last joint with non-zero torque. The external force, estimated after localization on a 4th link will intersect the axis of a 5th and 6th joint. By searching another collision point for these links we can see that there is another solution in a 6th link. From the current robot state, it not possible to decide which of them is a true contact location, everything we can do is to mark these points and see if changes in robot pose or external force will leave only one of them.

The real-time capabilities of the proposed method can be explained by the hierarchical robot surface representation and the fact that the most complicated part in estimation is the inversion of $3 \times 3$ matrix for each potential contact point. The most time-consuming part is a Jacobian calculation, but using an only $\boldsymbol{Jv}^{(i)}$ part of a (2) we need to once in a timestamp find the forward kinematics to each joint origin and extract $\boldsymbol{Z_j},\ \boldsymbol{O_j}$. As a result, this algorithm could be run in real-time.

## V. Experiments & Results

For the experimental study, the KUKA iiwa 14 collaborative robot was used. The obtained method was tested in simulation and hardware. For the simulation, we generated 1000 random robot poses and applied external force in 22 contact points giving 22000 samples in total. For the real robot, only 10 test configurations were used to verify our results. All points of contact were located in the last four links where a random force from a friction cone was applied. To match a contact point location on the real robot and our virtual model the ATOS 5 measurement system was used. 22 ATOS 5 markers were placed on the real robot surface to estimate their exact location with sub-millimeter accuracy. During the experiment on the real robot, the operator applied external force only in the marked point with a known location on the robot surface. This allowed us to obtain ground truth information about contact point $p_c$. The ground truth for force vector amplitude or direction was not provided, but in the future, we plan to use a special tool with a force sensor.

The developed algorithm was tested on Intel Core i5-4210H 3GHz CPU, 8Gb RAM PC as a Matlab program.

The result of virtual and real experiments presented in Table III. For our test, we used 5 different parameters for our algorithm and one case where we simply iterate through all vertices of robot mesh. The vertices on the mesh were located with a 0.5 cm mean distance between them. As a parameter, desired size of a cluster $d\_size$ from Algorithm 2 was set as 10, 25, 50, 100 vertices, that give approximately 500, 200, 100, 50 segments per link. In the simulation scenario, $N(0, 0.5Nm)$ noise was added to the torque values to emulate dynamic model inaccuracies in external torque estimation. The accuracy metrics in this table correspond to the average distance between the ground truth point of contact and the closest estimated local minimum from a set of possible contacts. The run-time evaluates the average time to estimate a contact point.

TABLE III
LOCALIZATION ALGORITHMS RESULTS FOR SIMULATED AND REAL ROBOT KUKA IIWA 14.

| Mesh parameters | Accuracy, cm | | Loop run-time | | |
|---|---|---|---|---|---|
| | Sim. | Real | ms | Hz | IF* |
| All points | 2.32 | 2.17 | 75 | 13.3 | - |
| Cluster $d\_size = 100$ | 3.19 | 3.04 | 1.8 | 555.6 | 41.6 |
| Cluster $d\_size = 50$ | 2.56 | 2.34 | 1.6 | 625 | 46.9 |
| Cluster $d\_size = 25$ | 2.48 | 2.33 | 1.5 | 666.7 | 50 |
| Cluster $d\_size = 10$ | 2.51 | 2.31 | 1.7 | 588.2 | 44.1 |

*Improvement Factor

TABLE IV
COMPARISON BETWEEN PROPOSED AND EXISTING LOCALIZATION ALGORITHMS.

| Algorithm | Robot | Acc., cm | Run-time, Hz | Mult. sol. |
|---|---|---|---|---|
| Proposed algorithm | KUKA iiwa | 2.3 | 600+ | Yes |
| Algorithm form [11] | KUKA iiwa | 4.2 | 100 | No |
| Feed-forward NN [6] | KUKA iiwa | 8.4 | 180 | No |
| Direct optimization [6] | KUKA iiwa | 5.4 | 19 | No |
| Contact PF [16] | KUKA iiwa | <2.4 | <10 | No |
| Proposed algorithm | Kinova Jaco2 | 1.8 | 600+ | Yes |
| DRL [7] | Kinova Jaco2 | 12 | 20 | No |
| RF [7] | Kinova Jaco2 | 8 | 200 | No |
| MLP [7] | Kinova Jaco2 | 4 | 200 | No |
| FCB [10] | Kinova Jaco2 | 11 | 63 | No |
| PFB [10] | Kinova Jaco2 | 12 | 71 | No |
| PFNN [10] | Kinova Jaco2 | 11 | 125 | No |
| PFWM [10] | Kinova Jaco2 | 11 | 159 | No |

According to the results, our algorithm allows achieving 50x speedup for the localization, without significant loss of accuracy. The model with $d\_size = 100$ shows worse performance due to a large cluster that is hard to represent with one reference point, which leads to picking the wrong segments and results in the loss of accuracy. Using a very small cluster $d\_size = 10$ requires evaluating more segments and slower due to increased cluster initialization time. The most optimal parameter is a $d\_size = 25$ since it gives a good balance between run-time and accuracy.

The accuracy of an algorithm depends on the link with a collision. The last links tend to have a better accuracy due to the larger number of sensors. With $d\_size = 25$, estimated accuracy in simulation with $N(0, 0.5Nm)$ noise is 0.9, 1.2, 2.2, 4.2 cm for the 7th, 6th, 5th, and 4th link respectively.

The proposed algorithm with $d\_size = 25$ was also compared to the existing localization algorithms for a KUKA iiwa and simulated Kinova Jaco 2 (with $N(0, 0.5Nm)$ noise for torque measurements) robots in Table IV. None of the existing approaches was able to find more than one possible solution. It should be noted that run-time results are achieved with a different computational power. Compared to algorithms based on a particle filter [8], [10], our approach shows better run-time frequency since we do not spend time on resampling and projecting points on surface steps. The absence of a numerical optimization procedure inside like Direct in [6], [7], or QP in [11], [8], [10] also helps to improve run-time.

## VI. CONCLUSION

In this work, we propose an algorithm for contact point localization on the surface of the robot. The algorithm is capable to detect more than one possible solution for the cases where external force produces zero torque on a joint. The estimation is based on the current robot state, using 1D joints torque sensors values, and a preprocessed robot mesh. The mesh preprocessing includes isotropic remeshing and a two-stage clustering to achieve a hierarchical structure. By exploiting this structure we achieve a real-time execution of our algorithm on frequencies more than 600 Hz. The approach was tested in the simulation on Kinova Jaco 2 and KUKA iiwa robot with an average accuracy of 1.8 and 2.5 cm. For the real KUKA iiwa robot, the average accuracy of a solution is 2.3 cm.

## REFERENCES


[1] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, and F. Leali, "Human-robot coexistence and interaction in open industrial cells," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101846, 2020.

[2] A. De Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *Proceedings of the IEEE international conference on robotics and automation*, 2005, pp. 999–1004.

[3] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[4] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[5] N. Likar and L. Žlajpah, "External joint torque-based estimation of contact information," *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 107, 2014.

[6] D. Popov, A. Klimchik, and N. Mavridis, "Collision detection, localization & classification for industrial robots with joint torque sensors," in *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 838–843.

[7] A. Zwiener, C. Geckeler, and A. Zell, "Contact point localization for articulated manipulators with proprioceptive sensors and machine learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 323–329.

[8] L. Manuelli and R. Tedrake, "Localizing external contact using proprioceptive sensors: The contact particle filter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5062–5069.

[9] J. Bimbo, C. Pacchierotti, N. Tsagarakis, and D. Prattichizzo, "Collision detection and isolation on a robot using joint torque sensing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[10] A. Zwiener, R. Hanten, C. Schulz, and A. Zell, "ARMCL - ARM Contact point Localization via Monte Carlo Localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[11] D. Popov and A. Klimchik, "Real-time external contact force estimation and localization for collaborative robot," in *IEEE International Conference on Mechatronics (ICM)*, vol. 1, 2019, pp. 646–651.

[12] ——, "Transfer learning for collision localization in collaborative robotics," in *Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, 2020, pp. 1–7.

[13] K. Crane, C. Weischedel, and M. Wardetzky, "Geodesics in heat: A new approach to computing distance based on heat flow," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 5, pp. 1–11, 2013.

[14] A. Jacobson *et al.*, "gptoolbox: Geometry processing toolbox," 2018, http://github.com/alecjacobson/gptoolbox.

[15] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted voronoi diagram," in *Computer graphics forum*, vol. 28, 2009, pp. 1445–1454.

[16] L. Manuelli, "Localizing external contact using proprioceptive sensors: The contact particle filter," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.